# Machine Learning with Graphs (MLG)

## Homework 1 Report

吳岱桀 E14051350

# 1 INTRODUCTION

## 1.1 Motivation

A In the world nowadays, a new data type ‘graph’ has proposed to formulate interaction among entities. Take stream platform like Netflix or Spotify for example, we may have the history of user interaction which is the record that user may watch or listen to certain movies or music, and we can convert the history data into user-item graph for the use of recommending system. Other applications such as friend recommendation, atomic connection in Chemistry and citation relationship in are also common aspect to make use of the graph.

In order to get insight from the content of these interactions behind the graph, it is important to have systematic method to analyze the property of the graph. With the advancement of the technology and computing resources, more and more new methods for graph analysis are available proposed in recent years. In this report, we’ll discuss a new approach to evaluate Betweenness Centrality of graphs on the basis of Graph Neural Network.

## 1.2 Betweenness Centrality [1]

Betweenness centrality measures the number of times a node lies on the shortest path between other nodes. This measure shows which nodes are ‘bridges’ between nodes in a network. It does this by identifying all the shortest paths and then counting how many times each node falls on one. Nodes with high betweenness may have considerable influence within a network by virtue of their control over information passing between others. They are also the ones whose removal from the network will most disrupt communications between other nodes because they lie on the largest number of paths taken by messages. It's used for finding the individuals who influence the flow around a system and is useful for analyzing communication dynamics.

Betweenness centrality finds wide application in network theory; it represents the degree to which nodes stand between each other. For example, in a telecommunications network, a node with higher betweenness centrality would have more control over the network, because more information will pass through that node. It also applies to a wide range of problems in network theory, including problems related to social networks, biology, transport and scientific cooperation.

## 1.3 Calculating Betweenness Centrality – The Challenge

As the era of IoT and the maturity of the hardware leading to a better capacity to store all kinds of data, the graph recording different types of data become extremely large that the nodes and edge have increased to the amount that people have never imagined before. Although there're research proposed to evaluate betweenness centrality previously, the accuracy of these algorithms decreases, and the execution time increases considerably along with the increase in the network size. Moreover, there are many cases requiring BC to be dynamically maintained, where the network topology keeps changing. In large-scale online systems such as social networks and user-item network, the computation of BC may not finish before the network topology changes again.

It's necessary to compute the shortest paths of all node pairs to obtain betweenness centrality. The best known algorithm for computing BC exactly is the Brandes algorithm [2], whose time complexity is O(|V | |E|) on unweighted networks and O(|V | |E| + |V |$^2_{\log}$|V |) on weighted networks, respectively, where |V | and |E| denote the numbers of nodes and edges in the network. Another algorithm to calculate betweenness centrality is Floyd–Warshall algorithm [3], whose time complexity is O (|V |$^3$). In the real-world application to calculate BC, it's hard to apply this algorithm on large graph like graph regarding the time complexity. Besides, once links between certain nodes no longer exist, it's unavoidable to compute the shortest paths and compute betweenness centrality of nodes with time complexity of O (|V |$^3$) because of the fact that this algorithm is based on the evaluating shortest path of global graph structure.

Overall, to overcome the issues of computing BC that allows it applying to the large-scale graph for the real-world application, two major problems should be reevaluated as follows:

(1) Time Complexity

(2) Accuracy

## 1.4 DrBC: A Novel Graph Neural Network Approach [4]

To solve the problems in the previous section, a DrBC framework based on the Graph Neural Network has been proposed. The goal of DrBC is to learn an inductive BC-oriented operator that is able to directly map any graph topology information into a

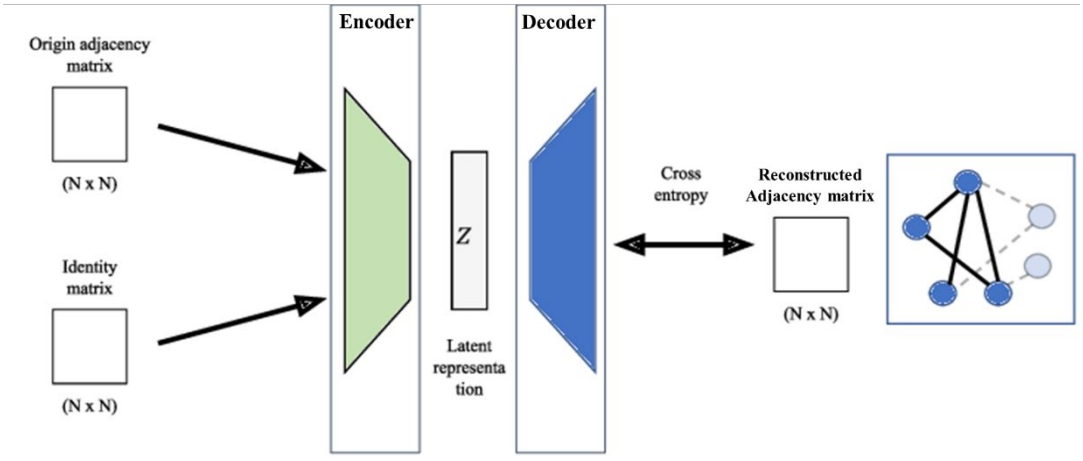
ranking score. Unlike previous algorithm to compute the exact BC value, the problem of identifying high BC nodes in DrBC has transformed into a learning problem.

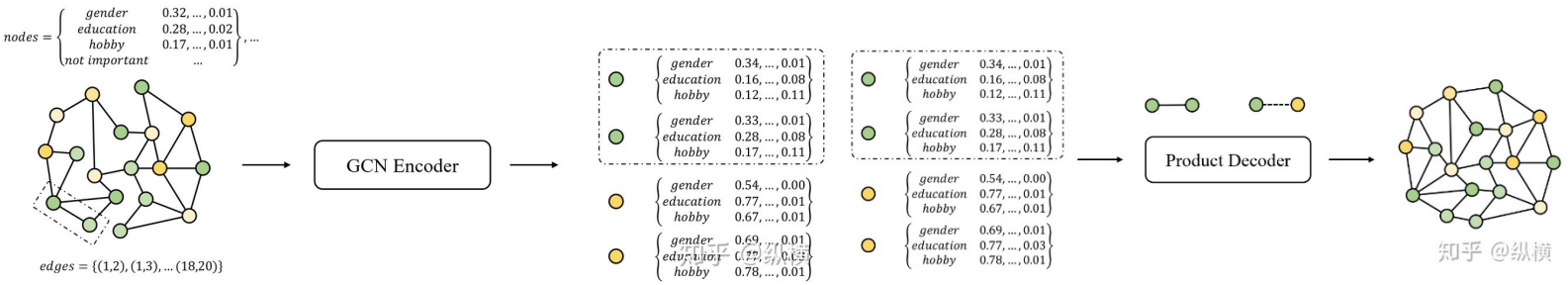
Instead of calculating approximating the exact BC value, the relative order of nodes with regard to BC is more important in DrBC.

In addition to taking another strategy getting relative ranking of BC, it takes an approach for predicting BC of target node based on information of the target node itself and information of the node's neighborhood that is different from exploiting global graph information to get the BC. An encoder-decoder framework is employing where the encoder maps each node to an embedding vector which captures the essential structural information related to BC computation, and the decoder maps embedding vectors into BC ranking scores. DrBC model predicts BC value locally and thus takes much fewer time to calculate, which makes DrBC suitable to be applied on large graphs behind internet platforms that's dynamically changing.before. Although there're research proposed to evaluate betweenness centrality previously, the accuracy of these algorithms decreases, and the execution time increases considerably along with the increase in the network size. Moreover, there are many cases requiring BC to be dynamically maintained, where the network topology keeps changing. In large-scale online systems such as social networks and user-item network, the computation of BC may not finish before the network topology changes again.

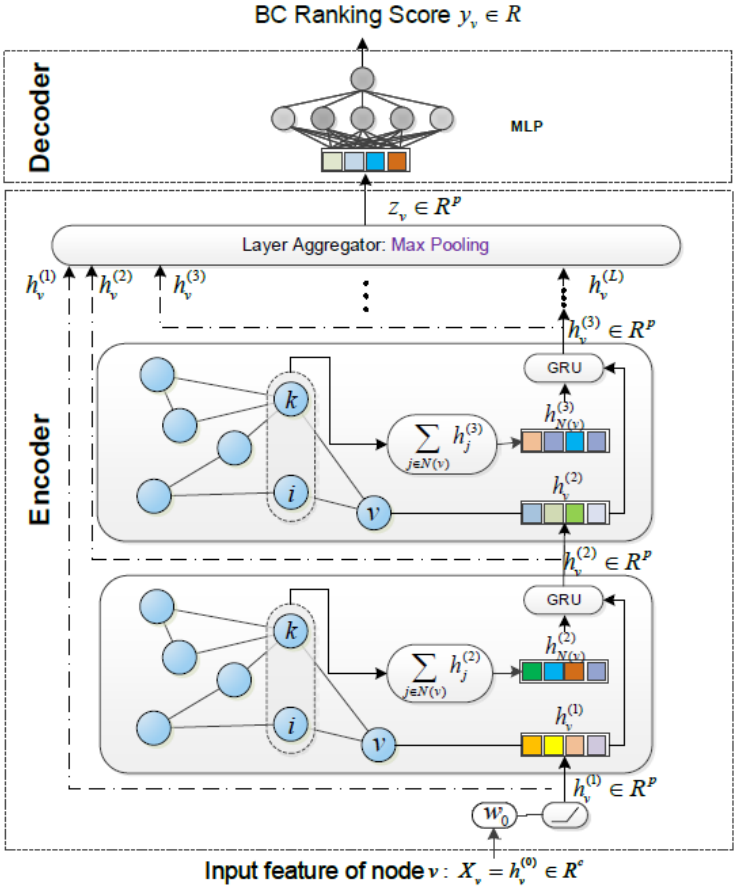
# 2 ALGORITHM AND ARCHITECTURE

## 2.1 Network Architecture






Inspired by Graph Auto Encoder (GAE), DrBC is adopting an encoder-decoder framework. A simple GAE is composed of two parts as illustrated in the above figure. The encoder part is a multi-layer Graph Convolutional Network (GCN) that yields the node embedding. As for the decoder, it takes inner-product operation on node embedding of a node pair. Cross Entropy is used as loss function to optimize the prediction task that output the probability of link status in the graph. The whole framework of DrBC is illustrated as follows:




The encoder in DrBC is responsible for aggregating node features, for example degree of each node in DrBC, and then combining the input features with neighbors nodes'

information to obtain the node embedding of each node. The encoder function is defined as follows, where A is adjacency matrix and X is node feature.

$$z_v = ENC(A, X; \Theta_{ENC})$$

Different from the object of decoder in a simple decoder that is to reconstruct the node structure information, the decoder in DrBC is to map the node embedding to the relative ranking BC scalar value using multilayer perceptron (MLP) on the other hand. The decoder function is defined as follows, where $W_5$ and $W_4$ are trainable weight.

$$y_v = DEC(z_v; \Theta_{DEC}) = W_5 ReLU(W_4 z_v)$$

Node degree is utilized as the initial input of encoder. The feature vector is defined as $[[d_v, 1, 1]]$, dv is the degree of the node. There's no detailed explanation of how two "1" values in the input vector are choose, but our speculation is that there's no specific reason and it can be replaced as other feature such as user property in the user-item graph of recommending system and the amount of publication in the citation graph. The initial feature vector is processed as initial node embedding $h_0$ after a single linear transformation operation with embedding size of 128. To get the embedding of next layer, Neighborhood Aggregation and COMBINE Function are needed. To aggregate the immediate neighborhood information of node v, the weighted sum aggregator is proposed to aggregate neighbors which is defined as follows:

$$h_{N(v)}^{(l)} = \sum_{j \in N(v)} \frac{1}{\sqrt{d_v + 1} \cdot \sqrt{d_j + 1}} h_j^{(l-1)}$$

Where j is neighbor node of node v, while $d_v$ and $d_j$ denote the degrees of node v and node j, $h_j^{(l-1)}$ denotes node j's embedding output by the $(l-1)$-th layer of the model.

The purpose of COMBINE function is to handle the combination of the neighborhood embedding generated by the current layer, and the embedding of the node itself generated by the previous layer. Different from employing sum and concatenation operation in previous works, GRU operation is adopting in DrBC. The neighborhood embedding $h_{N(v)}^{(l)}$ generated by the l-th layer is the input state, and node v's embedding $h_v^{(l-1)}$ generated by the $(l-1)$-th layer is the hidden state, then the embedding of node v at the l-th layer can be written as:

$$h^{(l)}_v = \text{GRUCell}\left(h_v^{(l-1)}, h_{N(v)}^{(l)}\right)$$

$$u_l = sigmoid(W_1 h_{N(v)}^{(l)} + U_1 h_v^{(l-1)})$$
$$r_l = sigmoid(W_2 h_{N(v)}^{(l)} + U_2 h_v^{(l-1)})$$
$$f_l = tanh(W_3 h_{N(v)}^{(l)} + U_3(r_l \odot h_v^{(l-1)}))$$
$$h_v^{(l)} = u_l \odot f_l + (1 - u_l) \odot h_v^{(l-1)}$$

The GRUCell is given as the above formulas where $\odot$ represents the element-wise product. As a part of RNN model, GRU utilizes the gating mechanism, which the update gate $u_l$ assists the model to decide how much information in the past needs to be kept, and the reset gate $r_l$ determines how much past information to be forgotten. It's available to adjust the proportion of the features of neighbors should be mixed with the local feature of each node with the help of GRU.

After node embedding of each layer is obtained, max operator is adopted to select the most informative layer for each feature coordinate. The Layer Aggregation function is defined as follows:

$$max(h_v^{(1)}, \ldots, h_v^{(L)})$$

The decoder is implemented with a two layered MLP which maps the embedding $z_v$ to the approximate BC ranking score $y_v$.

### 2.2 Optimization

Given a node pair (i, j), suppose the ground truth BC values are $b_i$ and $b_j$, DrBC predicts relative BC ranking scores $y_i$ and $y_j$ . Given $b_{ij} \equiv b_i - b_j$ , the goal is to keep the relative rank order of ground truth instead of exact value, it learns to predict $y_{ij} \equiv y_i - y_j$ with cross entropy loss function defined as follows:

$$C_{i,j} = -g(b_{ij}) * log\sigma(y_{ij}) - (1 - g(b_{ij})) * log(1 - \sigma(y_{ij}))$$

where $g(x) = 1/(1 + e^{-x})$. As such, the training loss is defined as:

$$Loss = \sum_{i,j \in V} C_{i,j}$$

It points out that they randomly sample 5|V | source nodes and 5|V | target nodes with replacement, forming 5|V | random node pairs to train the model.

# 3 EXPERIMENT & ANALYSIS

## 3.1 Evaluation

**Top-N%** accuracy is defined as the percentage of overlap between the top-N% nodes as returned by an approximation method and the top-N% nodes as identified by (ground truth):

$$\text{Top-}N\% = \frac{|\{\text{returned top-}N\%\text{nodes}\} \cap \{\text{true top-}N\%\text{nodes}\}|}{\lceil |V| \times N\% \rceil}$$

**Kendall tau** distance is a metric that calculates the number of disagreements between the rankings of the compared methods

$$K(\tau_1, \tau_2) = \frac{2(\alpha - \beta)}{n * (n-1)}$$

where α is the number of concordant pairs, and β is the number of discordant pairs

## 3.2 Comparison between Synthetic and Real-world graph

To generate the training data, the function powerlaw_cluster_graph in network is used with the following setting:

n, num of nodes: random(100,200)

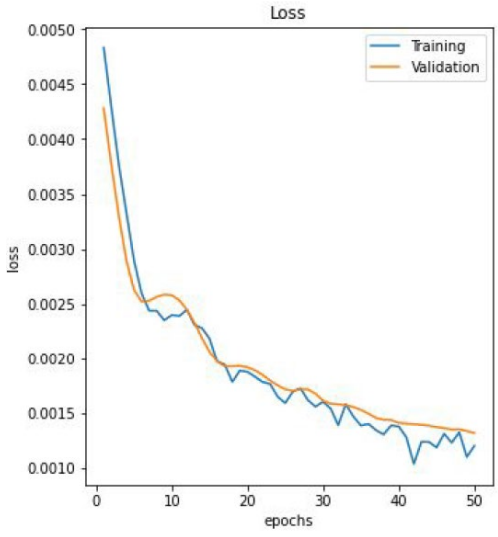
m, num of random edges to add each new node: 4

p, probability of adding a triangle after adding a random edge: 0.05

learning rate: 0.0001

batch size: 16

iteration: 10000 with early stopping



Synthetic:

| Data | Top-1% | Top-5% | Top-10% | Kendal | Time |
|---|---|---|---|---|---|
| DrBC | 0.846 | 0.858 | 0.834 | 0.853 | 0.4 |
| ABRA | 0.943 | 0.939 | 0.93 | 0.846 | 19.3 |
| RK | 0.937 | 0.931 | 0.929 | 0.795 | 18.6 |
| k-BC | 0.916 | 0.845 | 0.823 | 0.688 | 13.1 |
| KADABRA | 0.729 | 0.653 | 0.661 | / | 0.7 |
| Node2Vec | 0.168 | 0.197 | 0.228 | 0.009 | 35.8 |

Youtube:

| Data | Top-1% | Top-5% | Top-10% | Kendal | Time |
|---|---|---|---|---|---|
| DrBC | 0.597 | 0.588 | 0.601 | 0.514 | 415 |
| ABRA | 0.943 | 0.894 | 0.882 | 0.531 | 76321 |
| RK | 0.745 | 0.72 | ? | 0.142 | 119536 |
| KADABRA | 0.549 | 0.463 | 0.438 | / | 150 |
| Node2Vec | 0.136 | 0.179 | 0.211 | 0.419 | 5147 |

The results shows that there exists huge difference between 5k synthetic graph and YouTube graph data. The Top-10% accuracy of synthetic graph can achieve 83% while YouTube can only reach 60%. A rough speculation of the performance difference relies on the scale of the graph which synthetic data is consisted of 5k nodes while there're more than 100k nodes in the YouTube graph.

### 3.3 Comparison between Different n in Training Data

In this section, different size of node in training graph is compared as follows:

n: random(100, 200)

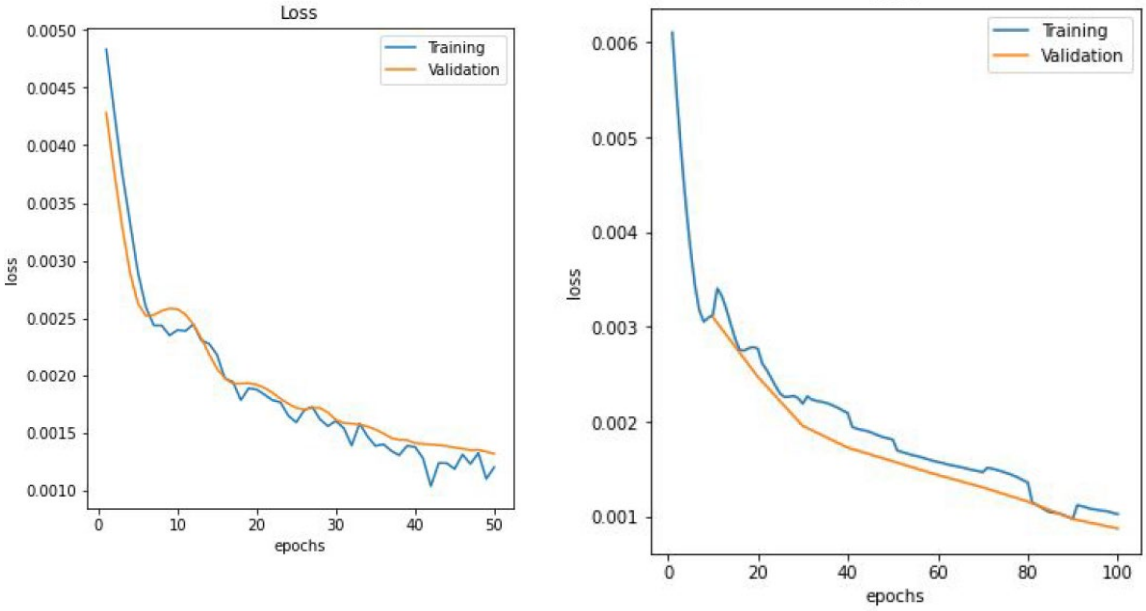
| Data | Top-1% | Top-5% | Top-10% | Kendal |
|---|---|---|---|---|
| 5k Synthetic Graph (Average) | 0.913 | 0.858 | 0.834 | 0.853 |
| Youtube | 0.597 | 0.588 | 0.601 | 0.514 |

n: random(50, 100)

| Data | Top-1% | Top-5% | Top-10% | Kendal |
|---|---|---|---|---|
| 5k Synthetic Graph (Average) | 0.908 | 0.851 | 0.826 | 0.839 |
| Youtube | 0.549 | 0.538 | 0.564 | 0.009 |

n: random(1000, 1200)

| Data | Top-1% | Top-5% | Top-10% | Kendal |
|---|---|---|---|---|
| 5k Synthetic Graph (Average) | 0.927 | 0.878 | 0.859 | 0.862 |
| Youtube | 0.635 | 0.613 | 0.601 | 0.55 |




The results shows that there the node number in training data has less impact for synthetic data than real-world data. The improvement in YouTube data is more significant as the node number increasing.

# 4 CONCLUSION

In this project we reproduce the framework of DrBC to compute relative ranking BC score. As compared with other method in previous work, it proves that this framework can deal with synthetic data and successfully meeting the requirement in both accuracy and time-efficiency goal, which confirms that GNN indeed helps the evaluating BC score by means of deep learning.

As the experiments shown in original work and the reproduction in this project, it may not perform well in certain domain of graph such as the data from YouTube, which leads to the concern of accuracy issue in the real-world application.

# 5 REFERENCES

[1] Betweenness centrality - Wikipedia

[2] Ulrik Brandes. 2001. A faster algorithm for betweenness centrality, The Journal of Mathematical Sociology, 25:2, 163-177, DOI: 10.1080/0022250X.2001.9990249

[3] Stefan Hougardy. 2010. The Floyd--Warshall algorithm on graphs with negative cycles. Inf. Process. Lett. 110, 8–9 (April, 2010), 279–281

[4] Fan, C., Zeng, L., Ding, Y., Chen, M., Sun, Y., & Liu, Z. 2019. Learning to identify high betweenness centrality nodes from scratch: A novel graph neural network approach. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management (pp. 559-568).